CLAIMS

What is claimed is:

1. A method of compiling code, comprising:

partitioning instructions in the code among a plurality of processors based on memory access latency associated with the instructions.

2. The method of Claim 1, wherein partitioning instructions comprises partitioning memory access dependence chains.

3. The method of Claim 1, wherein partitioning instructions comprises partitioning a memory access dependence chain into an upstream stage.

4. The method of Claim 1, wherein partitioning instructions comprises partitioning a memory access dependence chain into an upstream stage by assigning a first number of desired upstream nodes to the upstream stage, and assigning instructions in the code for which the first number of desired upstream nodes are dependent on to the upstream stage.

5. The method of Claim 4, wherein the number of desired upstream nodes is the length of the memory access dependence chain divided by a pipelining degree.

6. The method of Claim 4, further comprising:

generating a new number of desired upstream nodes if a compute weight of the upstream stage exceeds a predetermined value; and

assigning a first new number of desired upstream nodes to the upstream stage; and

assigning instructions in the code for which the first new number of desired upstream nodes are dependent on to the upstream stage.

7. The method of Claim 3, further comprising partitioning the memory access dependence chain into a downstream stage.

8. The method of Claim 7, wherein partitioning the memory access dependence chain into the downstream stage comprises:

assigning a last number of desired downstream nodes to the downstream stage; and

assigning instructions in the code which are dependent on the first number of desired downstream nodes to the downstream stage.

9. The method of Claim 8, wherein the number of desired downstream nodes is N*(d-1)/d, where N is a length of the memory access dependence chain, and d is a pipelining degree.

10. The method of Claim 1, further comprising identifying instruction dependence information.

11. The method of Claim 1, further comprising constructing a memory access dependence graph.

12. The method of Claim 1, further comprising:

constructing a memory access dependence graph; and

identifying a memory access dependence chain from the memory access dependence graph.

13. An article of manufacture comprising a machine accessible medium including sequences of instructions, the sequences of instructions including instructions which when executed cause the machine to perform:

partitioning instructions in code among a plurality of processors based on memory access latency associated with the instructions.

14. The article of manufacture of Claim 13, further comprising instructions which when executed causes the machine to further perform constructing a memory access dependence graph.

15. The article of manufacture of Claim 13, wherein partitioning instructions comprises partitioning a memory access dependence chain into an upstage stream by assigning a first number of desired upstream nodes to the upstream stage, and assigning instructions in the code for which the first number of desired upstream nodes are dependent on to the upstream stage.

16. A code partitioning unit, comprising:

a dependence information unit to identify dependencies between instructions in code; and

a code partitioning unit to partition instructions in the code among a plurality of processors based on memory access latency associated with the instructions.

17. The apparatus of Claim 16, wherein the code partition unit comprises:

a length unit to determine a number of desired lengths of upstream nodes to assign to an upstream stage;

an assignment unit to assign a first number of desired lengths of upstream nodes to the upstream stage;

a close up unit to assign instructions in the code for which the first number of desired length of upstream nodes are dependent on to the upstream stage; and

an evaluation unit to determine whether a compute weight of the upstream stage exceeds a predetermined value.

18. The apparatus of Claim 16, wherein the length unit determines a new number of desired length of upstream nodes in response to the evaluation unit determining that the compute weight of the upstream stage exceeds the predetermined value.

19. The apparatus of Claim 16, wherein the length unit determines a number of desired length of downstream nodes to assign to a downstream stage, the assignment unit assigns a first number of desired length of downstream nodes to the downstream stage, the close up unit assigns instructions in the code for which are dependent on the first number of desired length of down stream nodes, and an evaluation unit to determine whether a compute weight of the downstream stage exceeds the predetermined value.

20. The apparatus of Claim 19, further comprising a balancing unit to assign remaining instructions to the upstream stage and the downstream stage in a manner that substantially balances compute weight.